1-32228

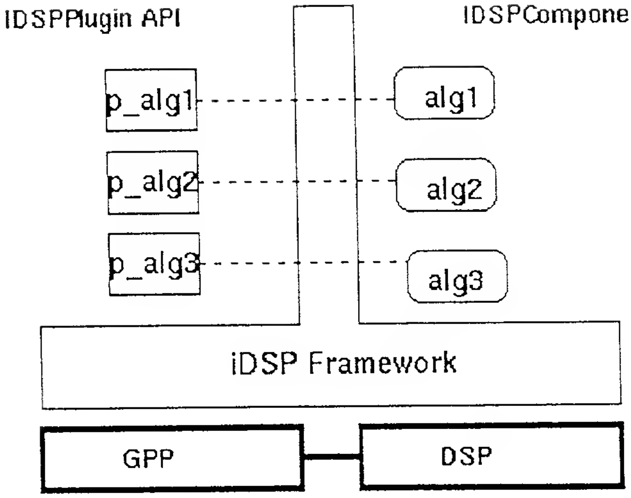


Fig. 1a

iDSP: Plug, Play, QoS

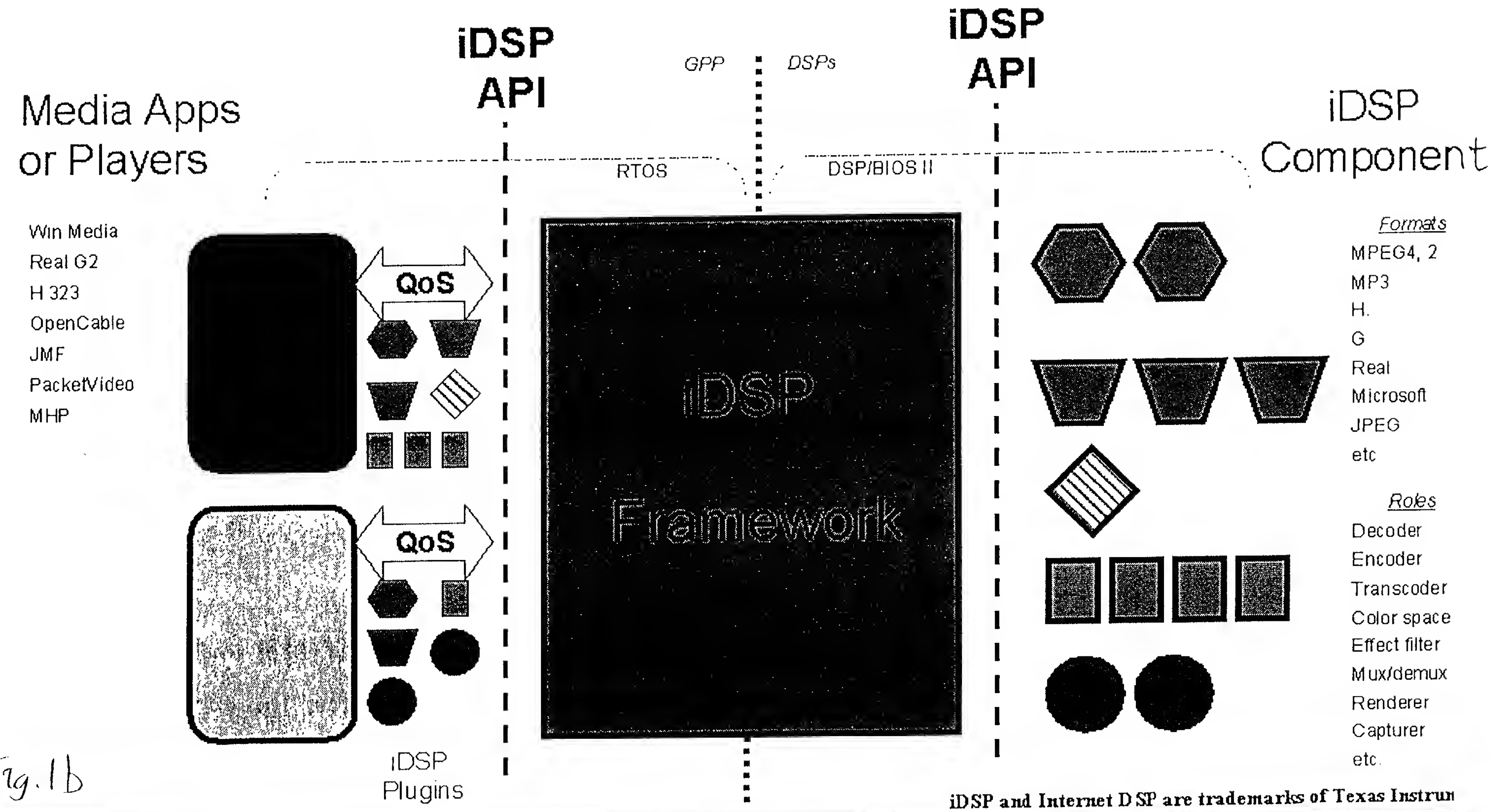


Fig. 1b

iDSP and Internet DSP are trademarks of Texas Instrun

iDSP Algorithm Chaining

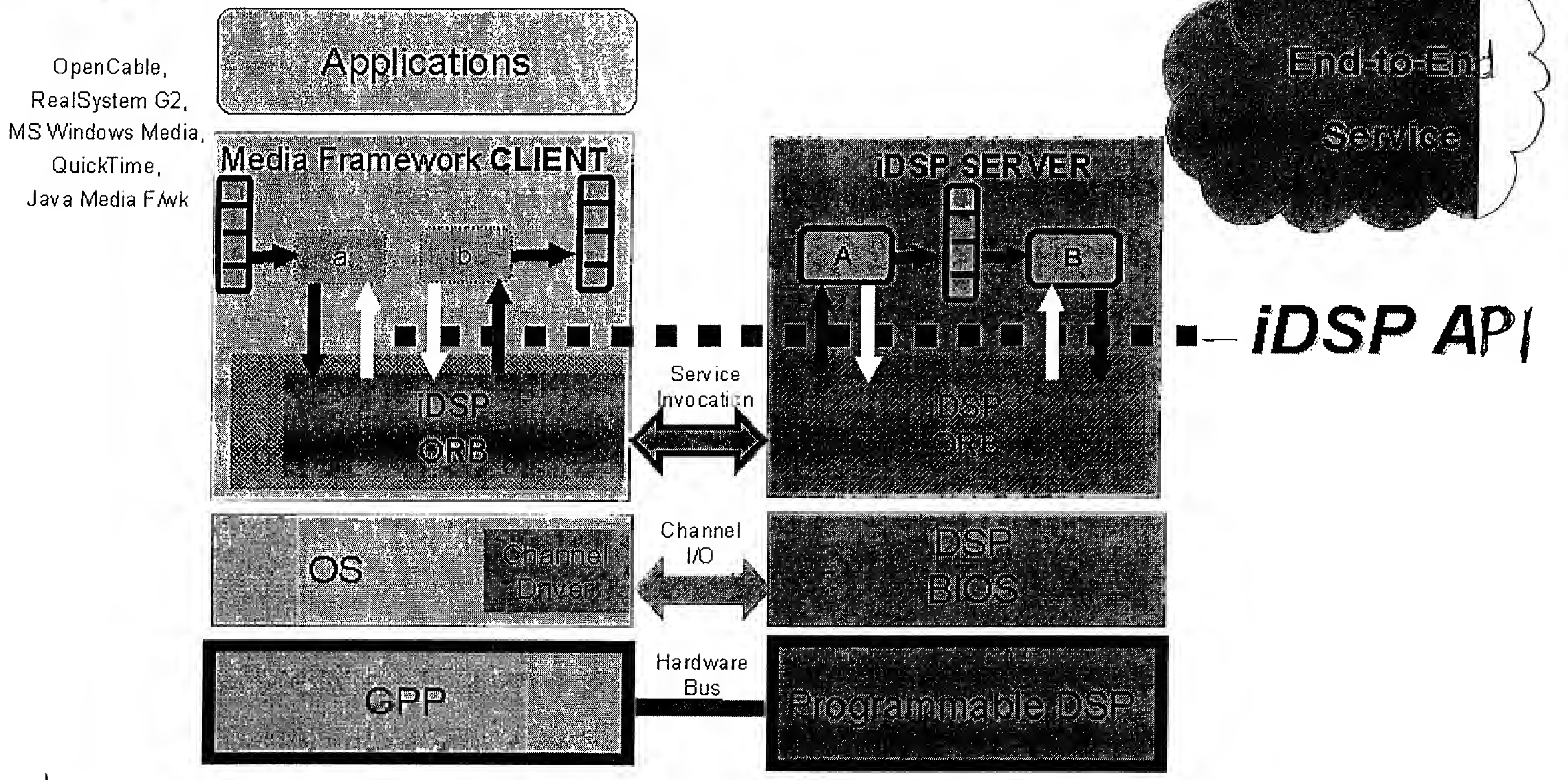


Fig. 1c

iDSP QoS Design

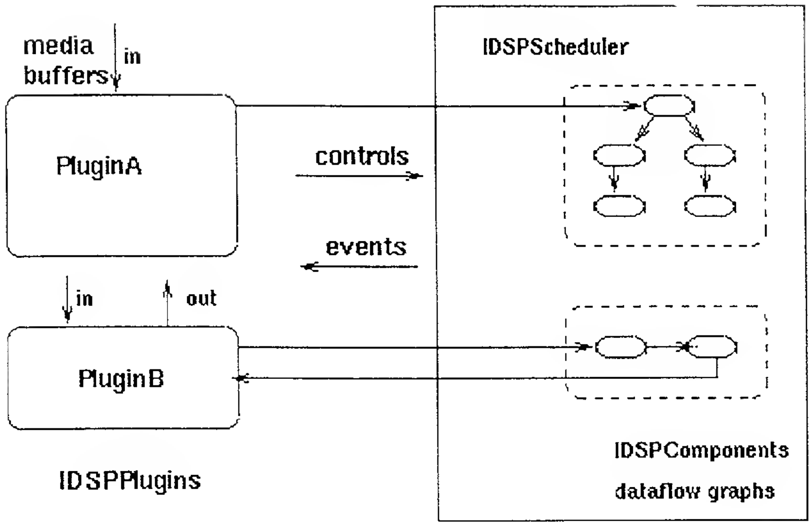


Fig 2a

IDSPQoSEvent types (received by IDSPPlugins)
* IDSPQoS_ALG_COMPLETED
* IDSPQoS_PRESENTATION_TIME_NOT_MET
* IDSPQoS_INSUFFICIENT_DATA
* IDSPQoS_INSUFFICIENT_CYCLES_AVAILABLE
* IDSPQoS_INSUFICIENT_MEMORY_AVAILABLE

IDSPQoSControl types:
* IDSPQoS_SET_RATE
* IDSPQoS_SET_QUALITY_LEVEL
* IDSPQoS_GET_STATS

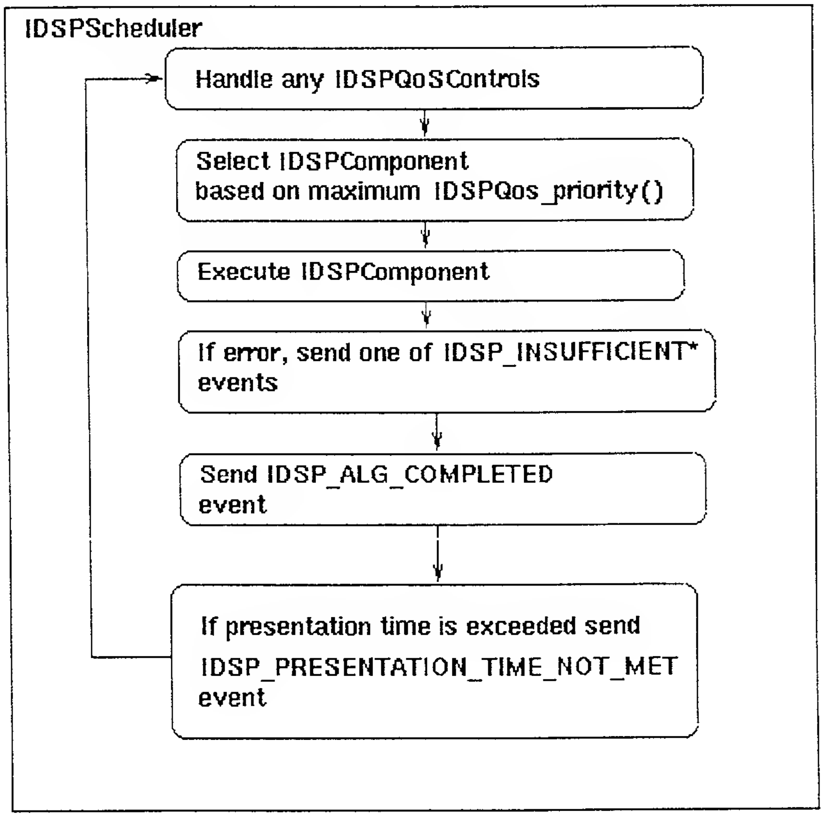

IDSPScheduler
Handle any IDSPQoSControls
Select IDSPComponent based on maximum IDSPQos_priority()
Execute IDSPComponent
If error, send one of IDSP_INSUFFICIENT* events
Send IDSP_ALG_COMPLETED event
If presentation time is exceeded send IDSP_PRESENTATION_TIME_NOT_MET event

Fig 2b

- The IDSPScheduler provides QoS scheduling and event notification:
  - IDSPQoS_priority() is computed based on the time-criticality to meet presentation deadline. If the highest priority component cannot be rum, the IDSPScheduler analyzes the evironent and sends an IDSPQoSEvent. The application can adjust the quality level or the rate.

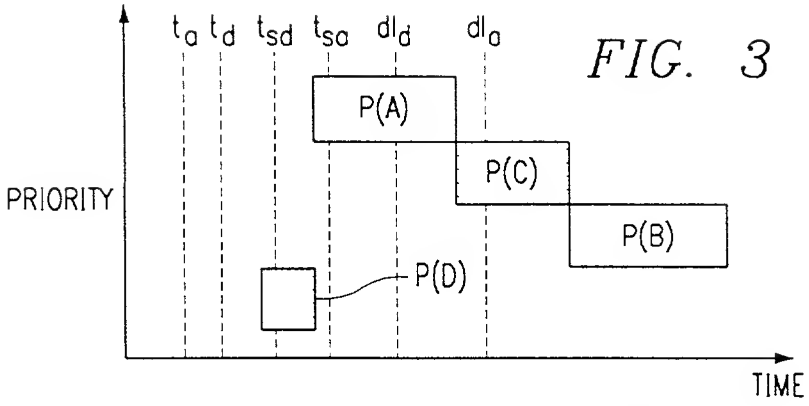


FIG. 3

t_{sa} = LAST POSSIBLE TIME FOR PROCESS A TO START AND STILL MAKES ITS DEADLINE

t_{sd} = LAST POSSIBLE TIME FOR PROCESS D TO START AND STILL MAKE ITS DEADLINE

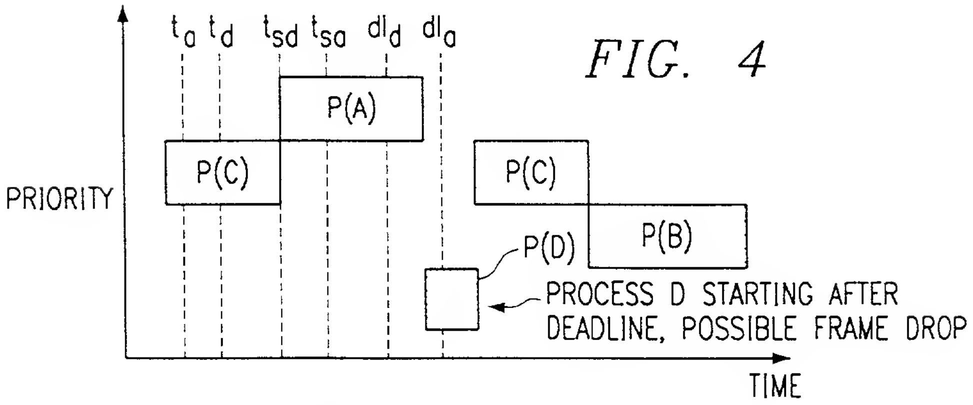


FIG. 4

t_{sa} = LAST POSSIBLE TIME FOR PROCESS A TO START AND STILL MAKES ITS DEADLINE

t_{sd} = LAST POSSIBLE TIME FOR PROCESS D TO START AND STILL MAKE ITS DEADLINE

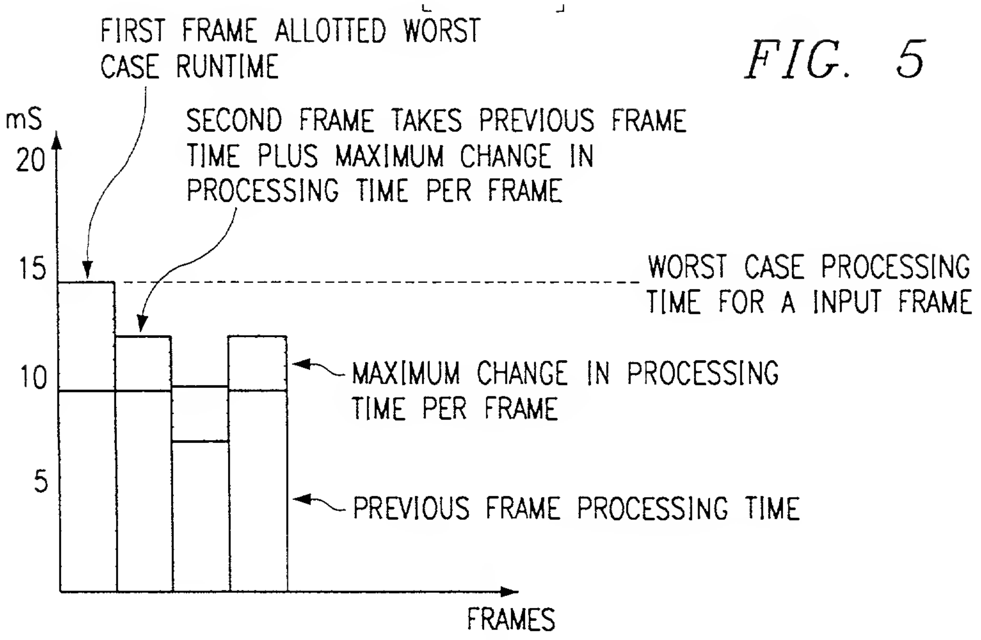


FIG. 5

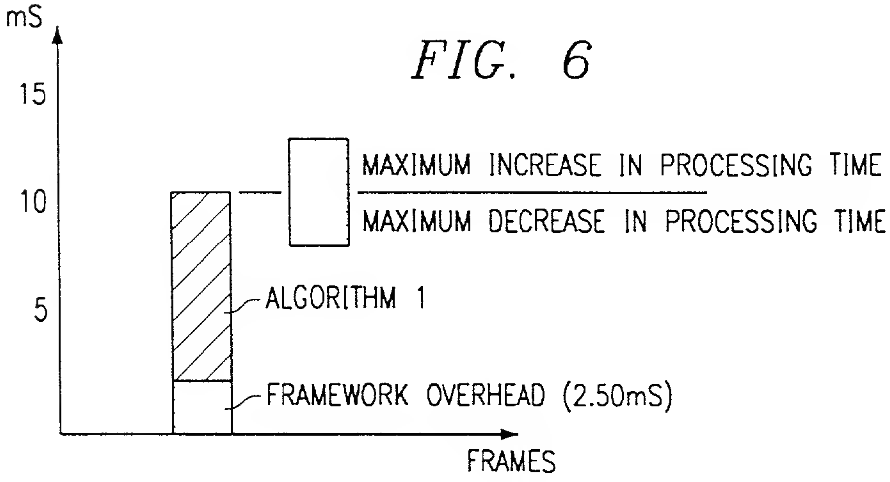


FIG. 6

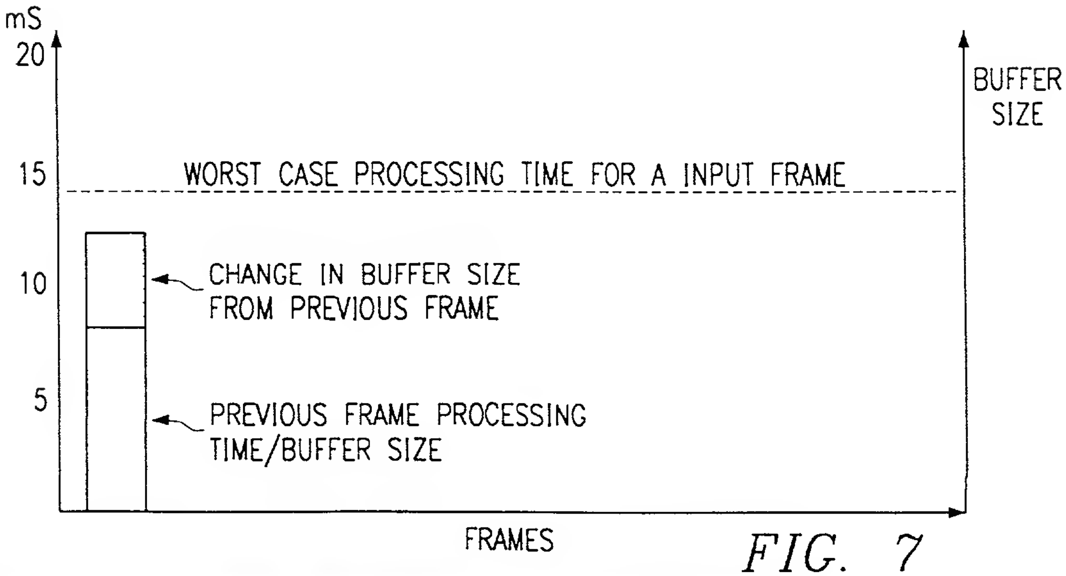


FIG. 7

FIG. 8

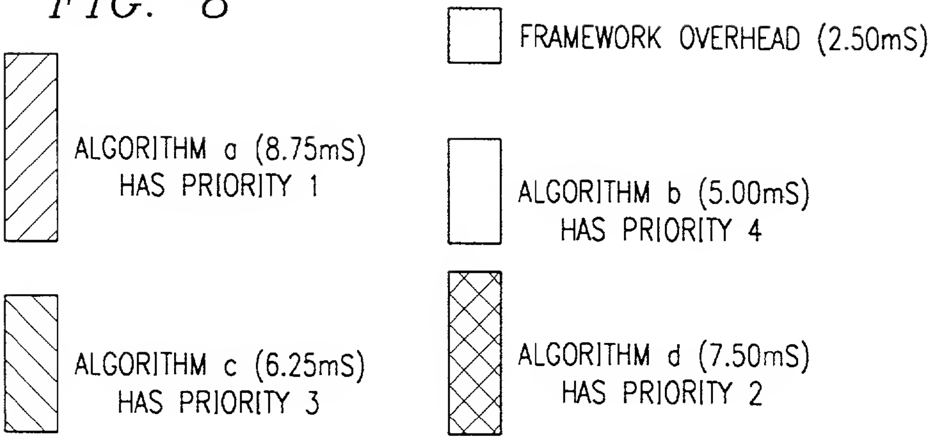
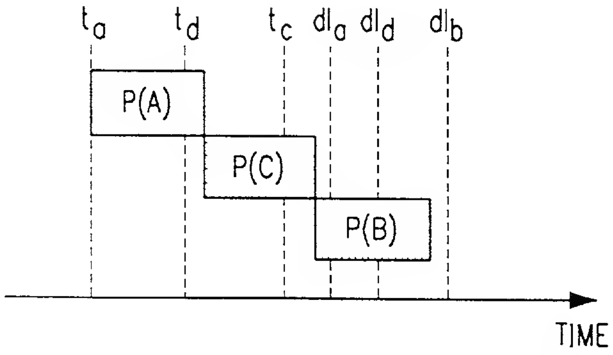

FRAMEWORK OVERHEAD (2.50mS)
ALGORITHM a (8.75mS) HAS PRIORITY 1
ALGORITHM b (5.00mS) HAS PRIORITY 4
ALGORITHM c (6.25mS) HAS PRIORITY 3
ALGORITHM d (7.50mS) HAS PRIORITY 2

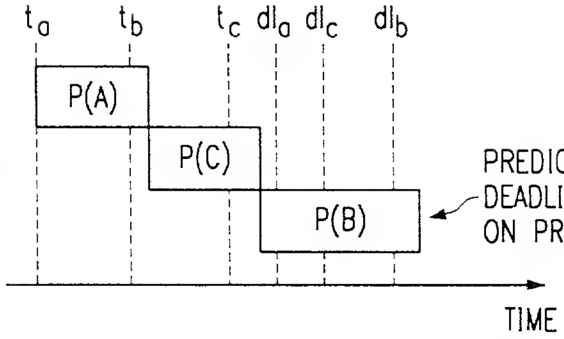
FIG. 9



t_i = TIME STAMP ARRIVAL OF EACH DATA FRAME FOR THE RESPECTIVE PROCESS

dl_i = DEADLINE FOR FINISHING PROCESSING OF EACH RECEIVED DATA FRAME

P() = PREDICTION OF PROCESSING TIME FOR EACH RECEIVED DATA FRAME



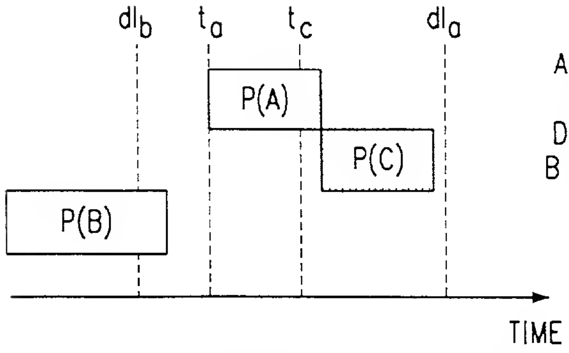
PREDICTION INDICATING DEADLINE WILL BE MISSED ON PROCESS B

FIG. 10

t_i = TIME STAMP ARRIVAL OF EACH DATA FRAME FOR THE RESPECTIVE PROCESS

dl_i = DEADLINE FOR FINISHING PROCESSING OF EACH RECEIVED DATA FRAME

P() = PREDICTION OF PROCESSING TIME FOR EACH RECEIVED DATA FRAME



BOTH PROCESS A AND C ARE PREDICTED TO COMPLETE BEFORE THEIR RESPECTIVE DEADLINES MEANING PROCESS B MISSING ITS DEADLINE DOES NOT RIPPLE THROUGH THE SYSTEM (YET)

t_i = TIME STAMP ARRIVAL OF EACH DATA FRAME FOR THE RESPECTIVE PROCESS

dl_i = DEADLINE FOR FINISHING PROCESSING OF EACH RECEIVED DATA FRAME

P() = PREDICTION OF PROCESSING TIME FOR EACH RECEIVED DATA FRAME

FIG. 11

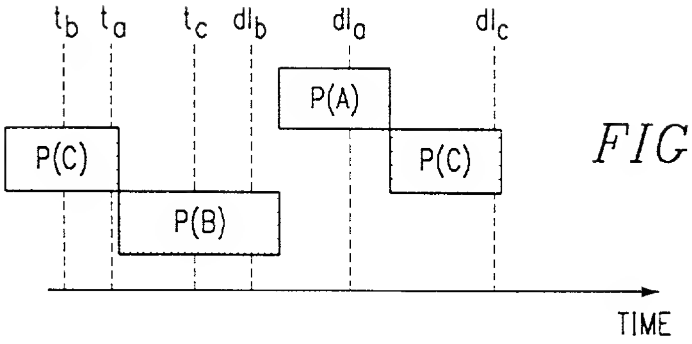


FIG. 12

t_i = TIME STAMP ARRIVAL OF EACH DATA FRAME FOR THE RESPECTIVE PROCESS

dl_i = DEADLINE FOR FINISHING PROCESSING OF EACH RECEIVED DATA FRAME

P() = PREDICTION OF PROCESSING TIME FOR EACH RECEIVED DATA FRAME

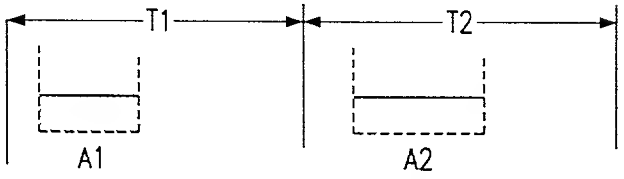


FIG. 13a

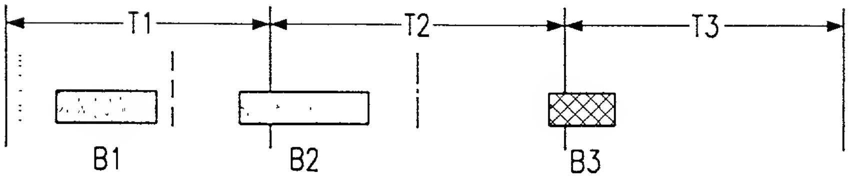


FIG. 13b

⋯⋯ ARRIVAL OF BUFFER B1

- - - ARRIVAL OF BUFFER B2

—·— ARRIVAL OF BUFFER B3

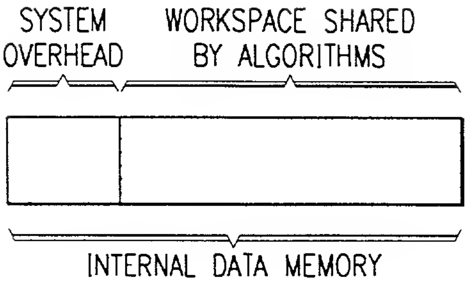


FIG. 14

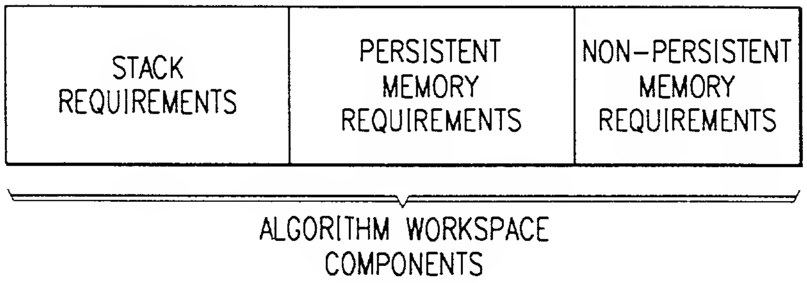

STACK REQUIREMENTS
PERSISTENT MEMORY REQUIREMENTS
NON-PERSISTENT MEMORY REQUIREMENTS
ALGORITHM WORKSPACE COMPONENTS

FIG. 15

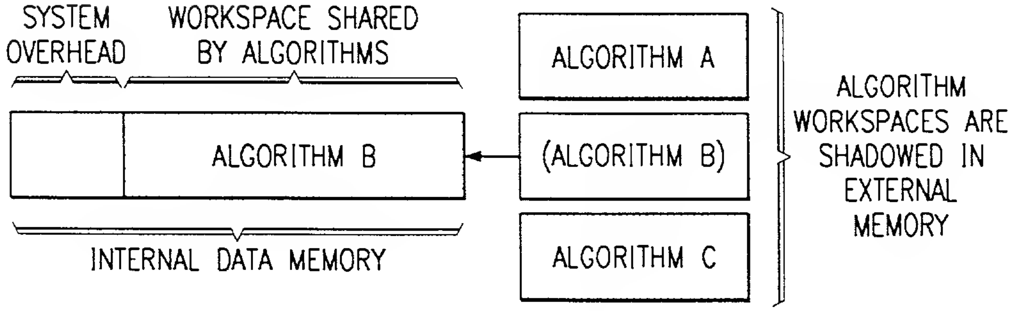

SYSTEM OVERHEAD
WORKSPACE SHARED BY ALGORITHMS
ALGORITHM B
INTERNAL DATA MEMORY
ALGORITHM A
(ALGORITHM B)
ALGORITHM C
ALGORITHM WORKSPACES ARE SHADOWED IN EXTERNAL MEMORY

FIG. 16

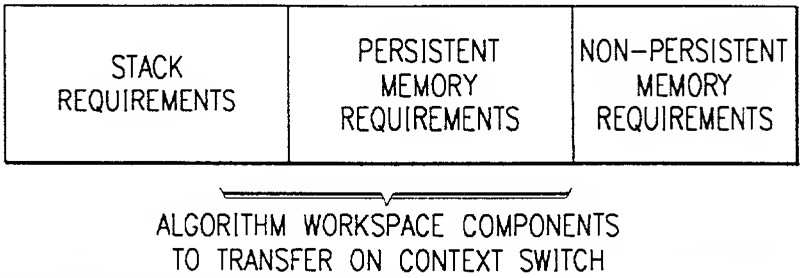
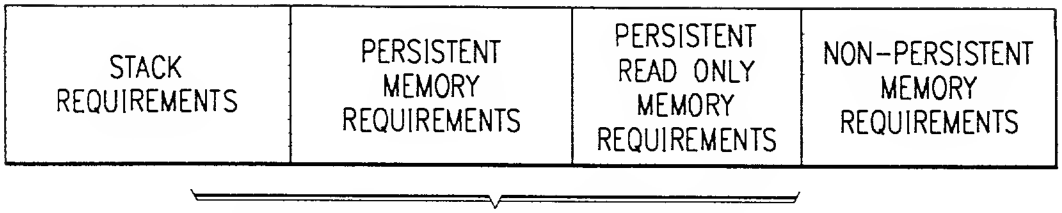
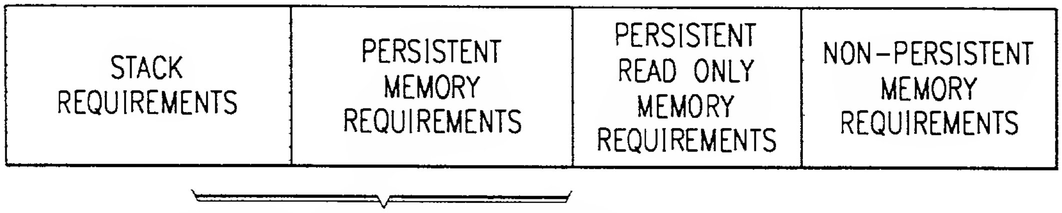

STACK REQUIREMENTS
PERSISTENT MEMORY REQUIREMENTS
NON-PERSISTENT MEMORY REQUIREMENTS
ALGORITHM WORKSPACE COMPONENTS TO TRANSFER ON CONTEXT SWITCH

FIG. 17


STACK REQUIREMENTS
PERSISTENT MEMORY REQUIREMENTS
PERSISTENT READ ONLY MEMORY REQUIREMENTS
NON-PERSISTENT MEMORY REQUIREMENTS

ALGORITHM WORKSPACE COMPONENTS TO TRANSFER IN PRIOR TO ALGORITHM EXECUTION IF ALGORITHM REQUIRES CONSTANT TABLES (CONTEXT SWITCH IN ONLY)

FIG. 18


STACK REQUIREMENTS
PERSISTENT MEMORY REQUIREMENTS
PERSISTENT READ ONLY MEMORY REQUIREMENTS
NON-PERSISTENT MEMORY REQUIREMENTS

READ ONLY PERSISTENT MEMORY DOES NOT NEED TO BE TRANSFERRED OUT ON CONTEXT SWITCH. THEREFORE ALGORITHM PAGE CHANGE-OUT IS MORE EFFICIENT.

FIG. 19

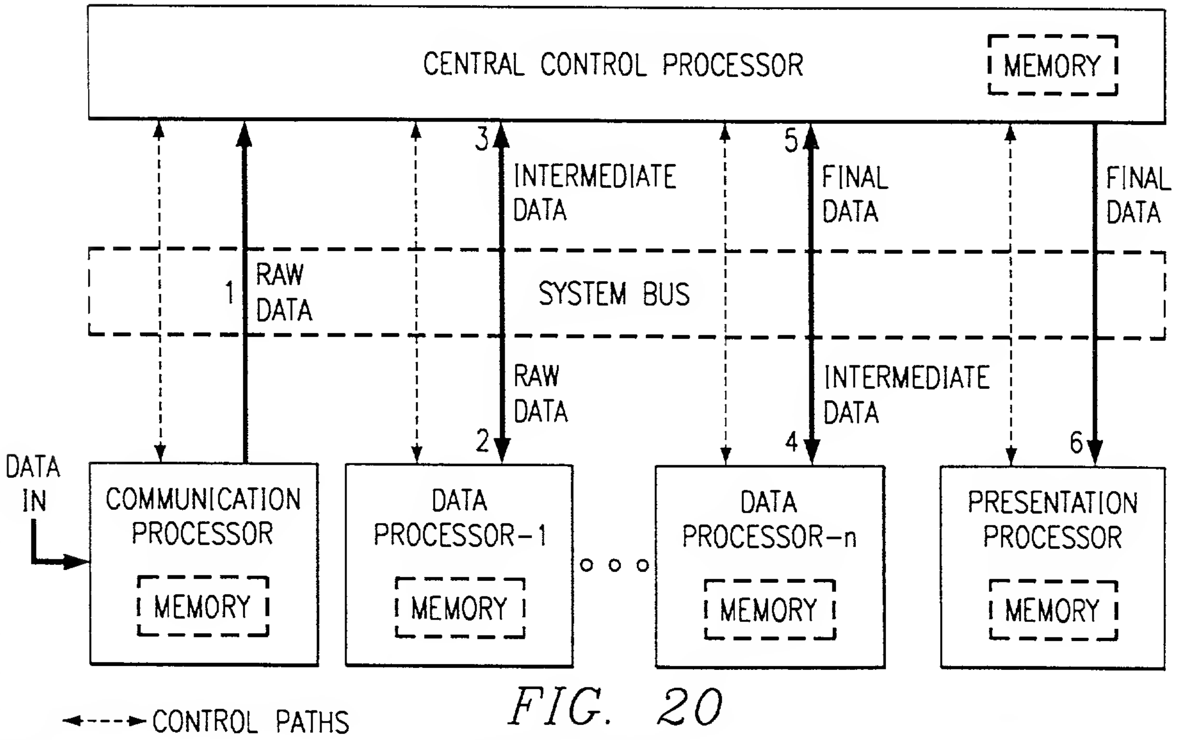


FIG. 20
(PRIOR ART)

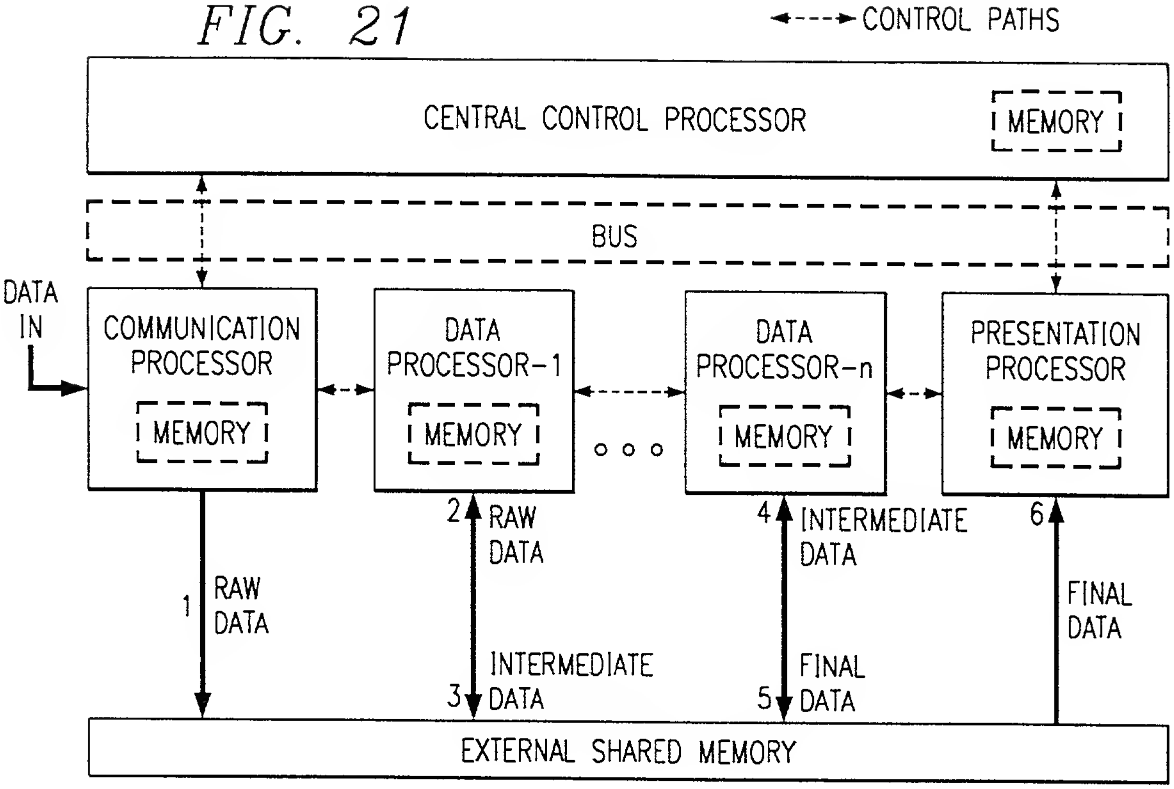

CONTROL PATHS
CENTRAL CONTROL PROCESSOR
MEMORY
BUS
DATA IN
COMMUNICATION PROCESSOR
MEMORY
DATA PROCESSOR-1
MEMORY
DATA PROCESSOR-n
MEMORY
PRESENTATION PROCESSOR
MEMORY
1
RAW DATA
2
RAW DATA
3
INTERMEDIATE DATA
4
INTERMEDIATE DATA
5
FINAL DATA
6
FINAL DATA
EXTERNAL SHARED MEMORY

FIG. 21

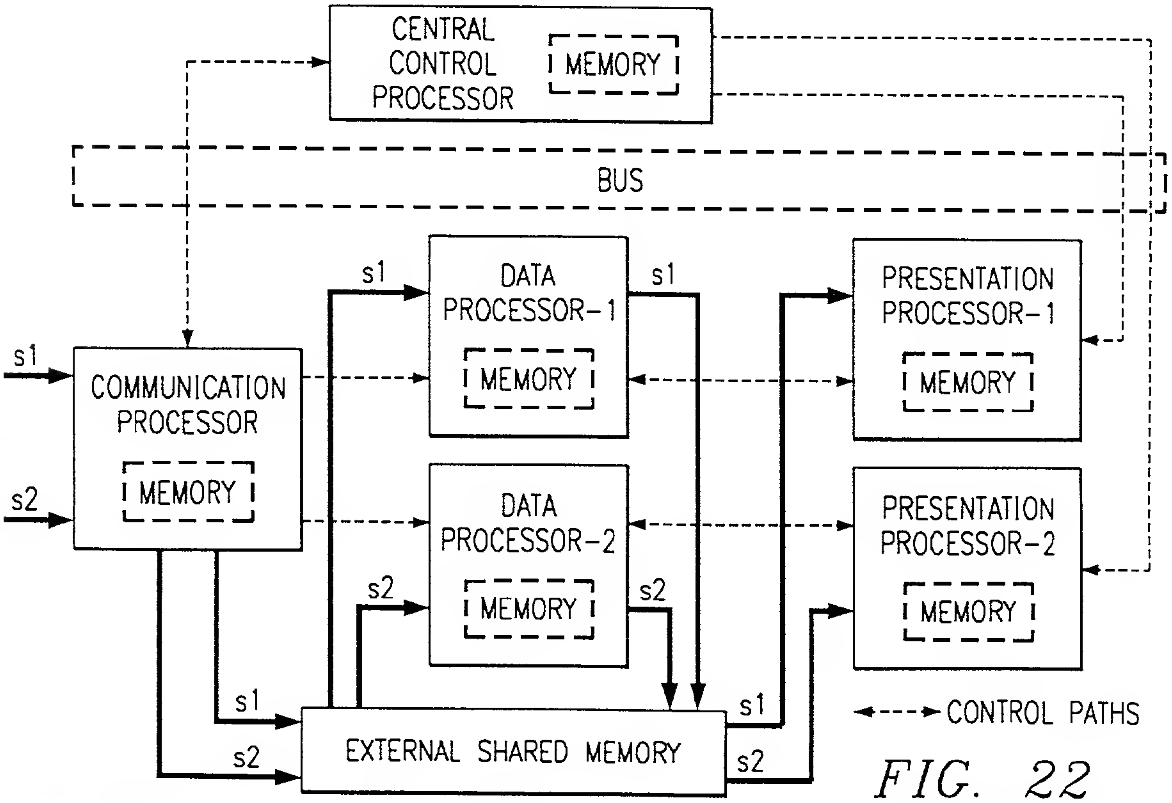
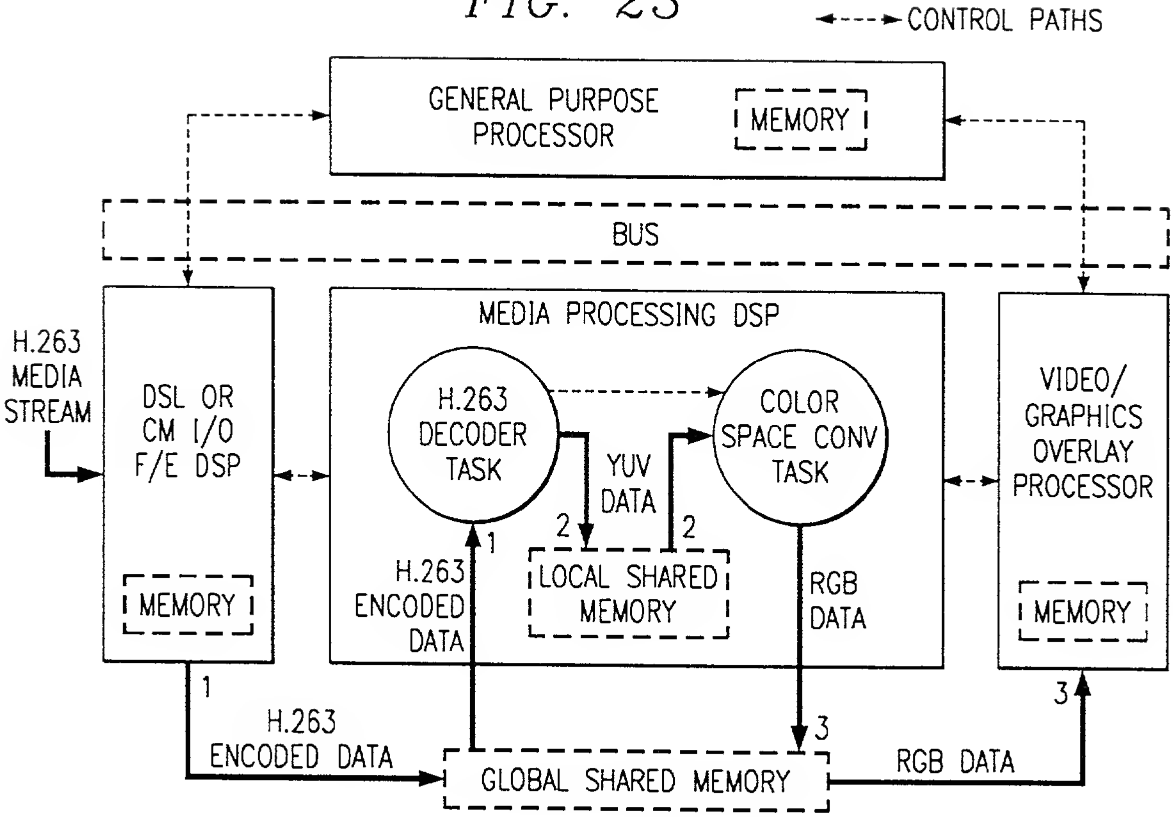

CENTRAL CONTROL PROCESSOR
MEMORY
BUS
s1
s2
COMMUNICATION PROCESSOR
MEMORY
s1
DATA PROCESSOR-1
MEMORY
s1
s2
DATA PROCESSOR-2
MEMORY
s2
PRESENTATION PROCESSOR-1
MEMORY
PRESENTATION PROCESSOR-2
MEMORY
s1
s2
EXTERNAL SHARED MEMORY
s1
s2
CONTROL PATHS

FIG. 22

FIG. 23


CONTROL PATHS
GENERAL PURPOSE PROCESSOR
MEMORY
BUS
H.263 MEDIA STREAM
DSL OR CM I/O F/E DSP
MEMORY
MEDIA PROCESSING DSP
H.263 DECODER TASK
COLOR SPACE CONV TASK
YUV DATA
2
2
1
H.263 ENCODED DATA
LOCAL SHARED MEMORY
RGB DATA
VIDEO/ GRAPHICS OVERLAY PROCESSOR
MEMORY
1
H.263 ENCODED DATA
GLOBAL SHARED MEMORY
3
RGB DATA
3

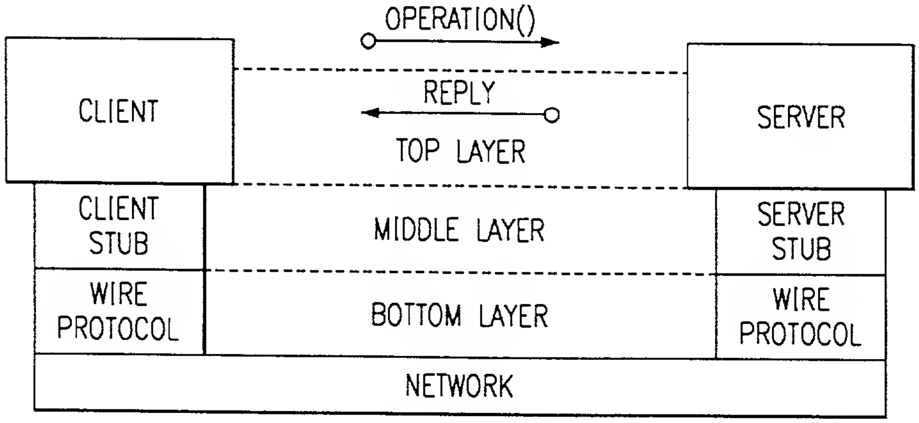

OPERATION()
CLIENT
REPLY
SERVER
TOP LAYER
CLIENT STUB
MIDDLE LAYER
SERVER STUB
WIRE PROTOCOL
BOTTOM LAYER
WIRE PROTOCOL
NETWORK

FIG. 24

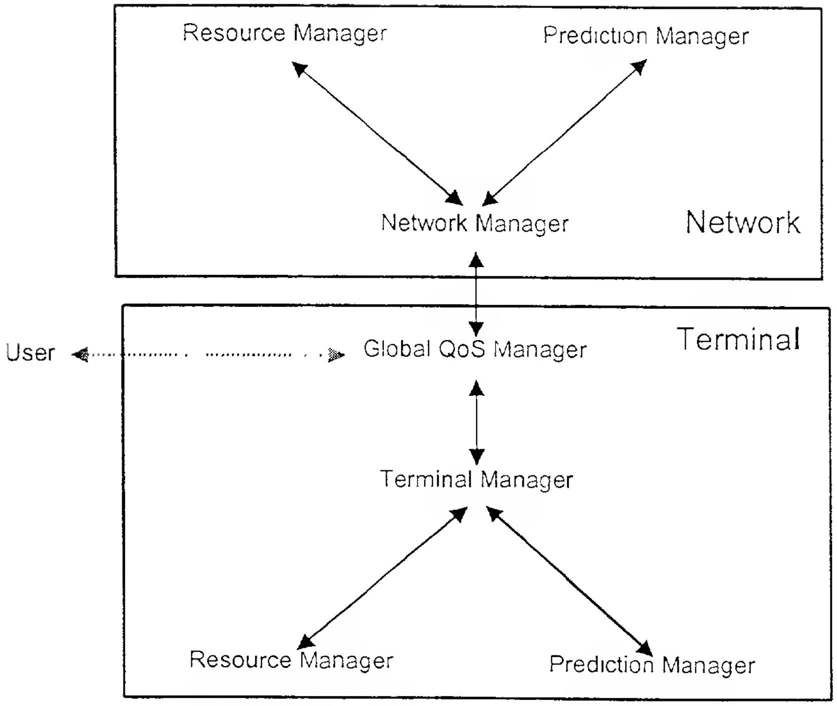


Figure 5: Simplified MPEG-21 Resource Management Framework

Figure25 shows a simplification of the proposed resource management framework, arrows denote control flow communication through APIs, not necessarily media flows. These control flows are governed by protocols. On the network side, these can coincide with

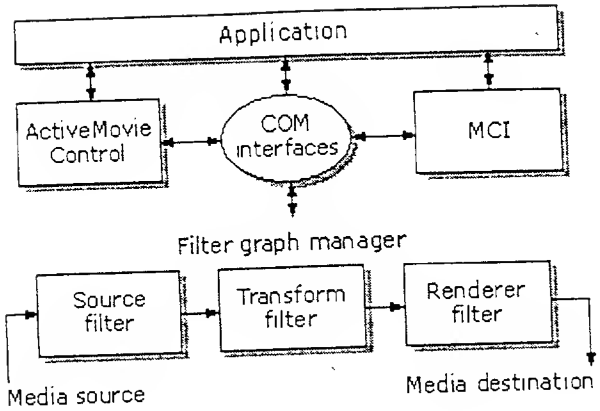


Fig 26

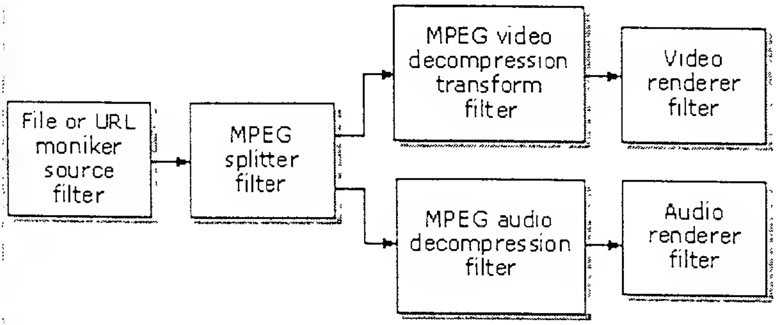

File or URL moniker source filter
MPEG splitter filter
MPEG video decompression transform filter
Video renderer filter
MPEG audio decompression filter
Audio renderer filter

Fig 27

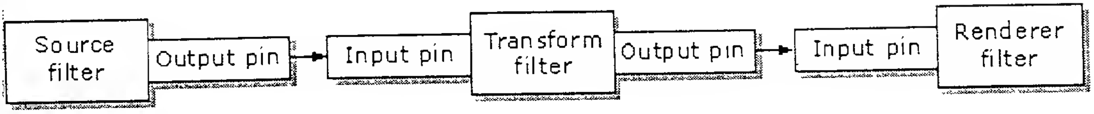


Fig 28